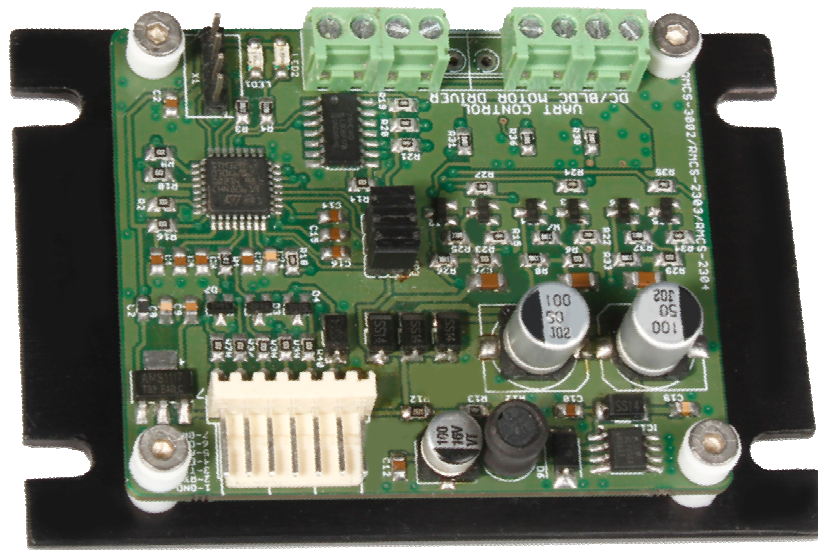


DC Servo Drive

10V - 30V DC, 50W with ASCII Modbus

RMCS – 2303



Operating Manual v1.0

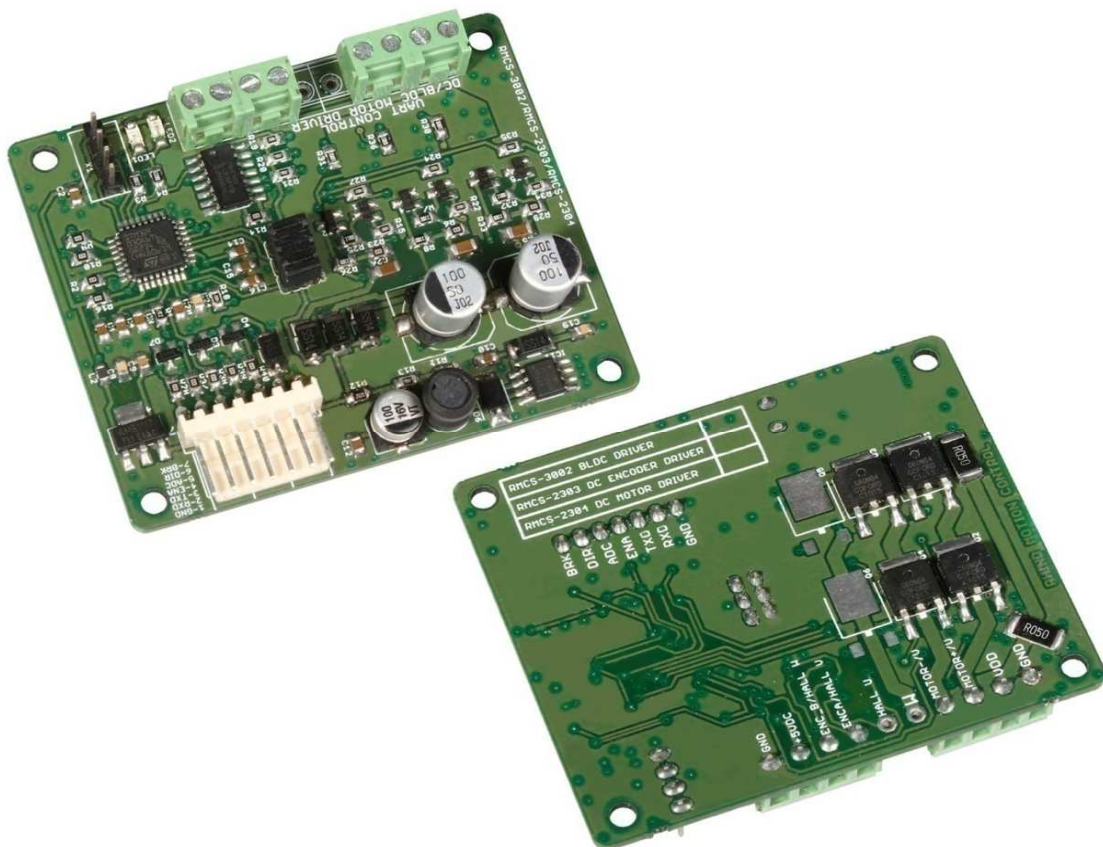
Contents

Introduction – Salient Features	3
Technical specifications and Pin description	4
Drive and motor connections	4
Modes of operation & Slave ID Addressing	6
Basics of servo drive	8
How to use the drive	9
Using PC Software.....	10
Installation of Software.....	10
COM Port Selection	12
Selection of Modbus Slave Address	13
Parameters - Read, Write, Save & Reset	14
Control Modes.....	16
Analog Control Mode	16
Digital Speed Control Mode.....	18
Position Control Mode	20
Modbus Registers	23
Brief Description of all Registers	24
Control Through Microcontroller (Arduino)	27
Hardware Connection	27
Library Installation.....	28
Description of all Functions	29

Introduction – Salient Features

Rhino Motion Controls RMCS-2303 with UART ASCII is a high performance dc servo drive (10–30 V DC) designed for optimized operation of any DC Servo motors with quad encoder feedback. This is an amazing cost effective solution to provide closed loop servo control for various applications. The salient features of this drive:

- This drive provides a closed loop speed and position control for DC motor with encoder.
- The motor programmed speeds are maintained irrespective of the voltages supplied.
- Also by using this drive, the rated torque of the motor is available at all speeds and the torque does not decrease with change in speeds.
- It is possible to run the motor in three different modes, Analog speed control mode, Digital speed control mode and Position control mode.
- It has short-circuit protection for the motor outputs, over-voltage and under-voltage protection and will survive accidental motor disconnects while powered-up.
- This drive is configured using MODBUS ASCII protocol via UART.
- There is a function in the drive for setting the Modbus Slave Address from 1 to 7 using physical jumpers (Hardware Setting) or using PC GUI or any other device like PLC or microcontroller which can provide proper data output on ASCII Modbus.
- There are three user modes in the drive :
 - Mode 0 - Analog Control Mode
 - Mode 1 - Digital Speed Control Mode
 - Mode 2 - Position Control Mode

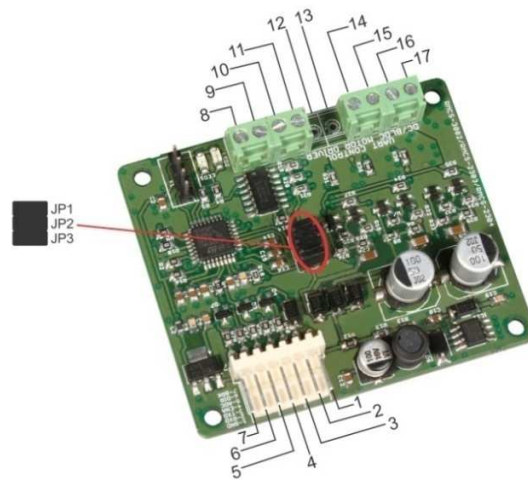


Technical specifications and Pin description

Supply Voltage and Current

Specification	Min	Max	Units	Comments
Supply Voltage	10	30	Volts DC	Between +Ve and GND
Phase Current	0.5	3	Amps	Peak 5 Amps per phase

Pin description of the drive is as per below image:



Pin No.	Description
1	GND
2	RXD
3	TXD
4	ENABLE
5	ADC
6	DIRECTION
7	BRAKE

(Pins 1-7 are used for drive Configuration and UART control)

Pin No.	Description
JP1	Jumper 1
JP2	Jumper 2
JP3	Jumper 3

(JP1 to JP3 are used for Hardware setting of slave ID)

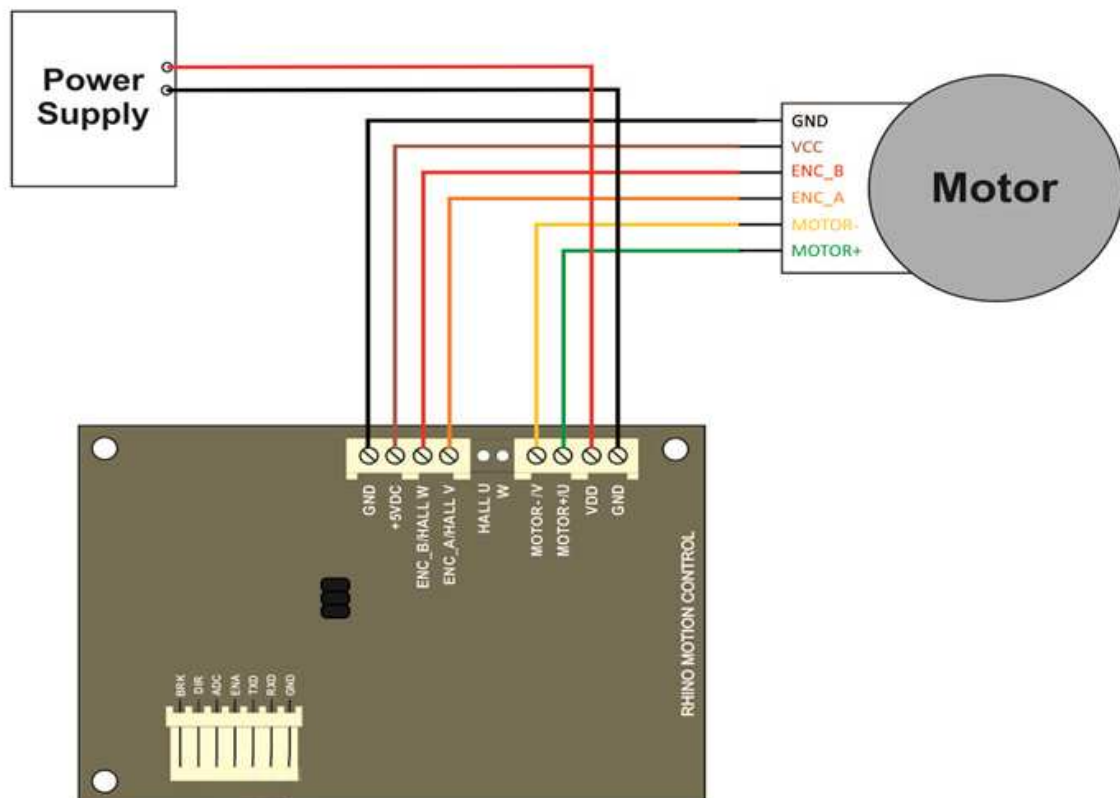
Pin No.	Description
8	GND
9	+5VDC
10	ENC_B/Hall W
11	ENC_A/Hall U
12	Hall U
13	W
14	Motor- / V
15	Motor+ / U
16	VDD
17	GND

(Pins 8 - 17 are connected to motor and power supply as described)

Drive & Motor Connection

Drive Pin outs		Motor Pin outs	
Pin No.	Description	Motor	Wire Color
8	GND	GND	Black
9	+5VDC	VCC(5 V DC)	Brown
10	ENC_B/Hall W	ENC_B(Encoder B)	Red
11	ENC_A/Hall U	ENC_A(Encoder A)	Orange
12	Hall U	M-(Motor-)	Yellow
13	W	M+(Motor+)	Green
14	Motor- / V		
15	Motor+ / U		
16	VDD		
17	GND		

Power Supply Pin outs	
VCC – 10 to 30V	
GND	



Modes of Operation

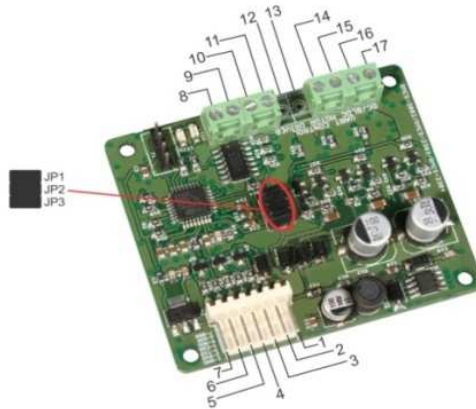
The drive can be configured in the following three modes

1. **Mode 0 Analog speed control mode :**
 - a. In this mode the speed of the Rhino DC Servo motor can be controlled by an externally connected Potentiometer.
 - b. Speed can be increased or decreased manually based on requirement using potentiometer.
 - c. The drive will provide full torque at all speeds within the range.
 - d. However the potentiometer has to be connected via a voltage divider to provide a maximum of 3.3 volts so as to not damage the drive.
 - e. Also the Enable, Brake and direction pins have to be connected as per configuration requirements.
2. **Mode 1 Digital Speed control mode with direction control :**
 - a. In this mode the speed and direction of the DC servo motor is settable / controllable via a Computer or any microcontroller board like Arduino or PLC or any other Modbus ASCII compatible device.
 - b. Like the analog mode here there is no compromise in the torque output of the motor irrespective of the operational speed and voltage supply and control at higher speeds.
 - c. This mode can be used when multiple motors are to be used to run at exactly the same RPM and same torque even though the voltage supply might be different.
 - d. Also in this mode the direction of the motor can be controlled digitally via Modbus ASCII commands to run the dc servo motor in both directions
 - e. In applications like conveyor belts where speed control is critical and industrial robotic applications like solar cleaning robots where straight line motion is critical for correction operation of the equipment, the digital speed control mode can provide optimal results.
3. **Mode 2 Position Control Mode :**
 - a. In this mode Speed, Direction, Acceleration and Position of motor can be controlled.
 - b. The position control mode is suitable in applications where exact movement of motor is required. This can be used in precision applications like machine control, motion control or robotics.

In all modes absolute encoder position and speed is available on Modbus.








Slave ID Addressing

Multiple drives with different slave addresses can be used with a single controller on single UART bus. For this all drives should have a unique Slave Address. Using the GUI software or Modbus commands, the slave ID can be set from 1 to 247. However, slave ID from 1 to 7 can be set using physical jumpers also. As shown in the image below there are three jumpers marked by a red circle shown in the image below.



The three jumpers JP1, JP2 and JP3 can be set in the configuration as per the below table to provide a physical slave address to the drive. In the below table a value of '0' corresponds to a state where the jumper is not connected and a value of '1' corresponds to a state where the jumper is connected. If none of the jumpers are connected the drive has been programmed to use the Slave ID 11 in default mode which can be changed from 1 to 247 using GUI software or Modbus commands.

Drive will check jumpers on startup. If no physical jumper is connected it will use programmed slave id.

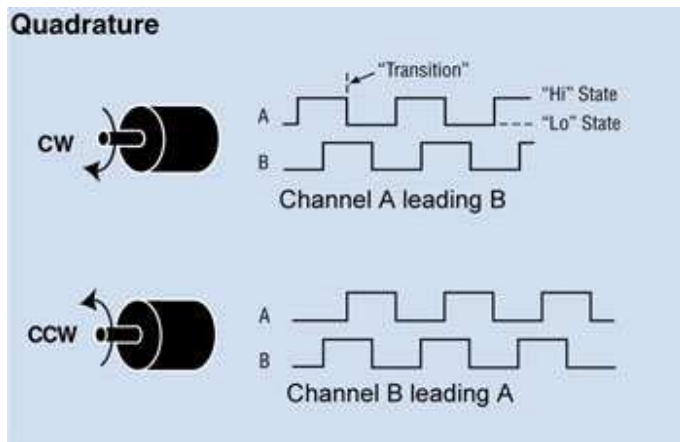
Slave ID	JP1	JP2	JP3	Image of connection on the Drive
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Basics of a servo drive

This is a servo drive to work with quad incremental encoder and dc motor. When the drive starts up the encoder position is set as 0. With reference to the start position drive will always have a 32 bit number keeping track of movement. It can be set to 0 at any point.

This servo drive works with quadrature encoder. The quadrature encoder also known as quad encoder or incremental encoder gives two channels of pulses output. It can be of optical or magnetic type.

The basic concept is the encoder gives output of two channels of pulse streams say channel A and channel B to drive, when A is leading B or B is leading A the drive can also determine the direction of movement.



So drive can calculate the speed and direction of motor according to the pulses if number of Lines per Rotation is known. For each line drive gets 4 counts, for example if encoder is 334 lines drive can get 1336 (334×4) counts per rotation. This parameter can be set in register 10. It's a 16 bit unsigned number having range of 1 to 65535.

In any of the 3 modes drive manages speed, direction, acceleration, deceleration, position of motor and voltage and current given to the motor based on set parameters, gains and feedback from encoder. This is a close loop system means in position mode the drive always tries to make the error which is difference between the required position and error to 0, in case of speed mode it tries to make the speed same as set speed.

The voltage and current given to motor is also managed by drive to provide higher torque when required. This also means that even at lower speed, higher torque up to the maximum torque available in motor is possible.

How to use the drive:

The drive needs to be configured to run in any one of the three modes described in the above section to make a closed loop control system. The drive can be configured using a PC with GUI software / Modbus controller / Arduino Board.

To configure the drive for a closed loop system

- PC with GUI software for Rhino 2303 drive or any generic Modbus software (Like Modbus Poll)
 - GUI software
<https://robokits.co.in/downloads/Rhino%20DC%20Servo%202303%20Config%20Setup.exe>
 - Modbus Poll demo version - <https://www.modbustools.com/download.html>
- RMCS-2303 UART ASCII Encoder DC Motor Driver
 - <https://robokits.co.in/motor-drives-drivers/encoder-dc-servo/rhino-industrial-encoder-dc-motor-driver-50w-with-uart-ascii-compatible-10-to-30-v-10a>
- Encoder DC Servo Motor (Any DC motor 10-30VDC 50W max with encoder)
 - <https://robokits.co.in/rhino-planetary-encoder-dc-servo>
 - <https://robokits.co.in/rhino-ig32-precision-dc-servo>
 - <https://robokits.co.in/motors/encoder-dc-servo/high-torque-dc-encoder-motor>
 - <https://robokits.co.in/motors/encoder-dc-servo/high-torque-high-precision-motor>
- Industrial Power Supply (below is a recommended supply. It can vary as per your requirements)
 - <https://robokits.co.in/power-supply/industrial-power-supply/24v-10a-industrial-power-supply>
- USB UART Module
 - RKI-1154 CP2102 - <https://robokits.co.in/control-boards/interface-boards/cp2102-usb-uart-module>
- External 10K Potentiometer and resistors (for analog Speed Control mode only).

Software Installation

Software is available here

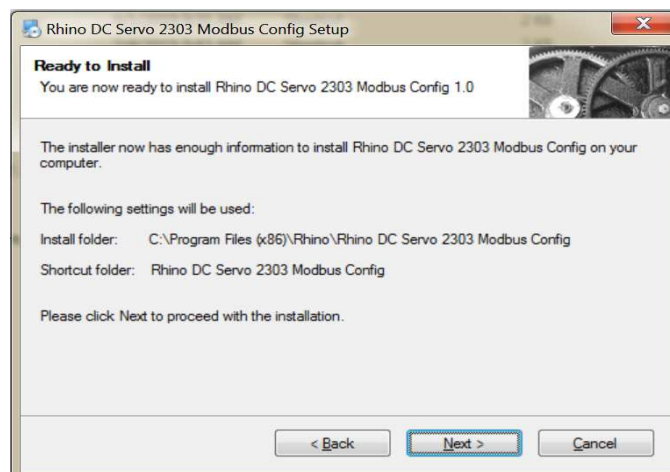
<https://robokits.co.in/downloads/Rhino%20DC%20Servo%202303%20Config%20Setup.exe>

Download and run the setup executable.

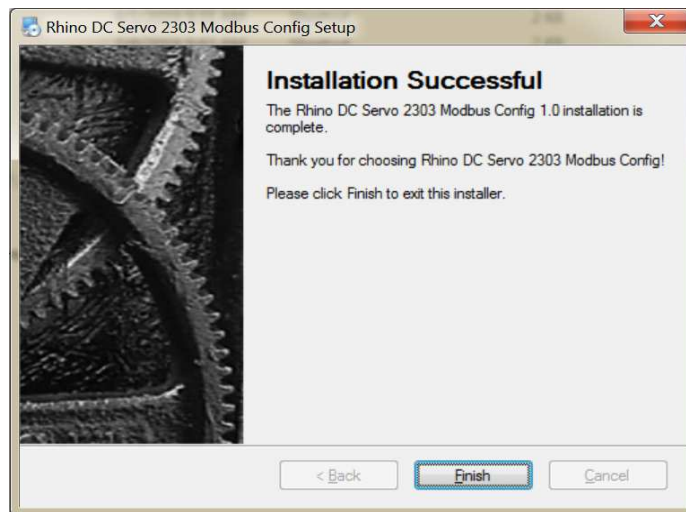
- Click **Next** button



- Verify installation destination and click **Next**

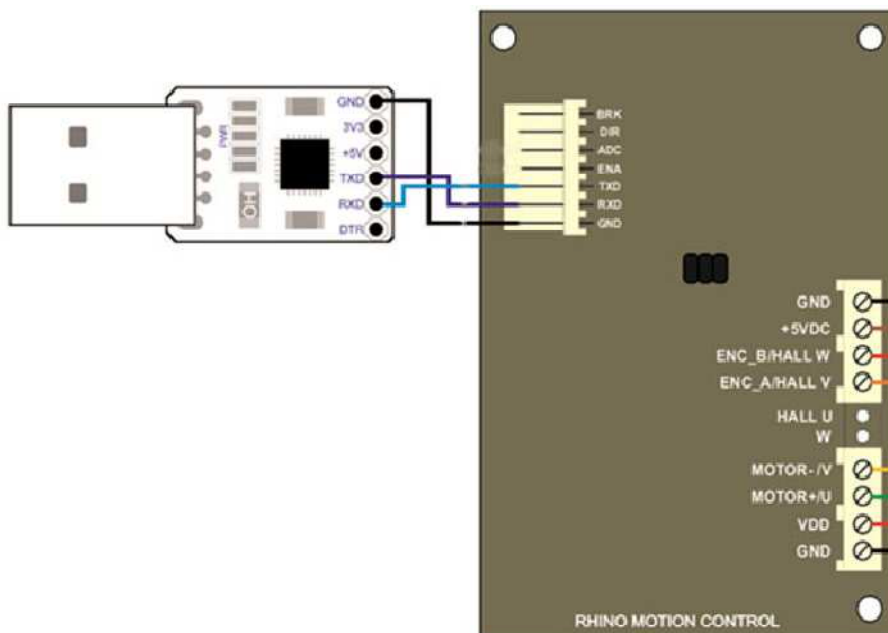


- Click Finish when Installation Successful message shows up.



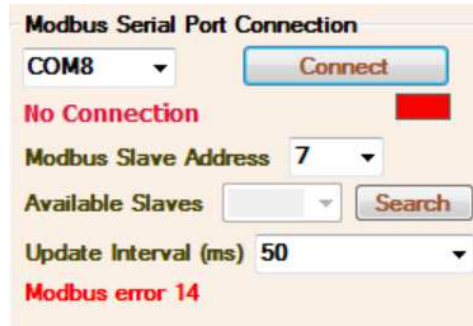
Now, Software is ready to use. find the shortcut on desktop and open the software.

Once the Software is installed connect USB-UART as per shown in below figure.

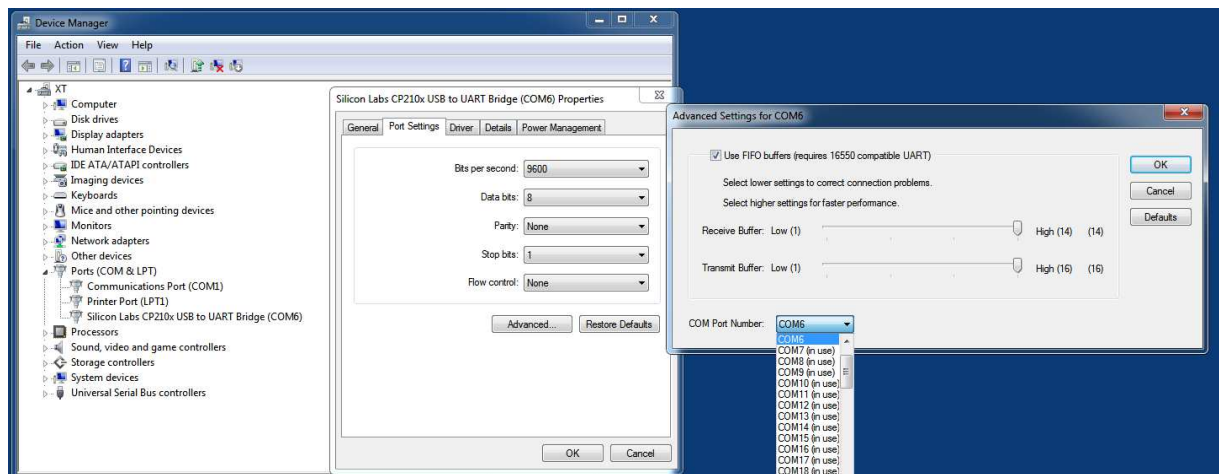


COM port Selection

Select COM Port for USB-UART , then click on **"Connect"** button.

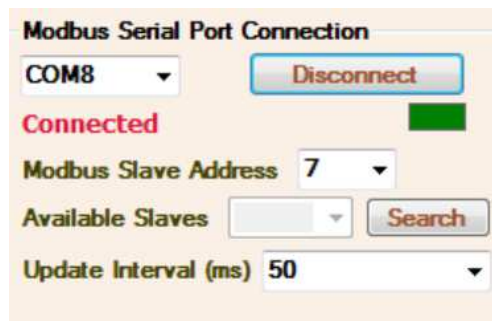


You may look in 'Devices and Printers' or 'Device Manager' for specific serial port of your device. The software supports serial ports up to COM32 only. If your device's port number is higher you can change it in - Device Manager > Ports (COM & LPT) > Double click on device name > Port Settings > Advanced > COM Port number



Once your drive is connected to COM Port and it shows **"Connected"** message.

When the drive communicates with software You can see a blinking Green indication.



Selection Of Modbus Slave Address

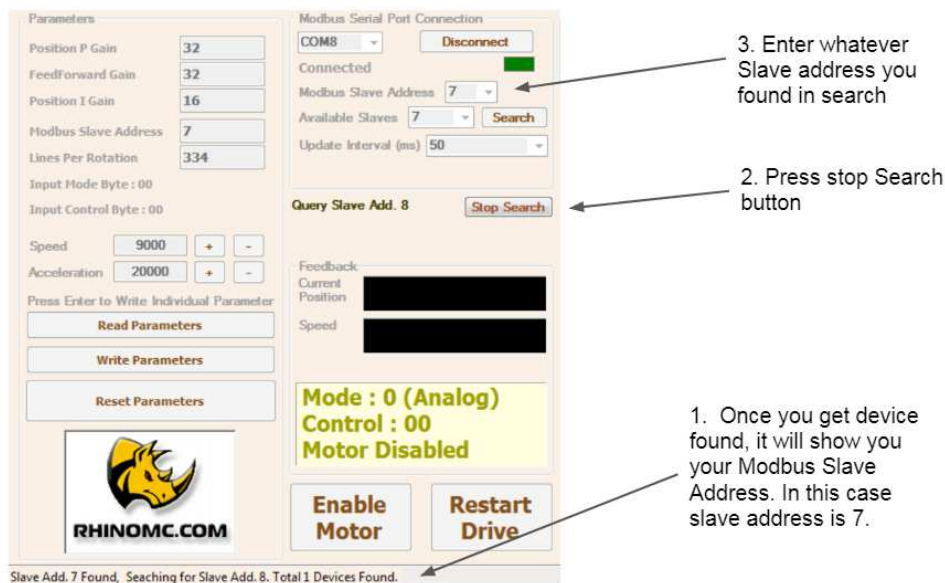
There are two ways to select Modbus Slave address.

1. If Slave address is already known, it can be directly selected from drop box.



2. If slave address is not known or multiple drives with multiple addresses are connected on serial bus, click search button. This will search for all slave ids from 1 to 247. If any response is received from slave id it will give indication in status bar and list the drive in 'Available Slaves' drop down box.

To search slave address click on "Search". Once all the connected devices are found click on "Stop Search". Select any of detected slave address in Modbus Slave Address drop down box and drive will start communicating to software.



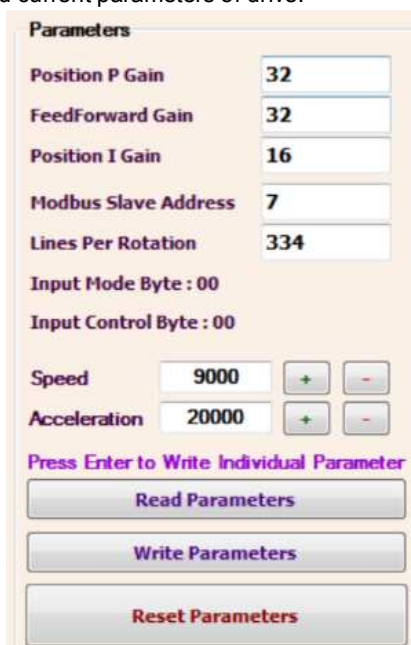
Make sure no multiple drives with same slave address is connected on serial bus.

Once any drive is connected and slave id for the same is selected, software will read parameters of the drive. It can also be done manually by clicking 'Read Parameters' button.

Parameters

READ PARAMETERS

"Read Parameters" is used to read current parameters of drive.



The screenshot shows a window titled "Parameters" with the following fields and controls:

- Position P Gain: 32
- FeedForward Gain: 32
- Position I Gain: 16
- Modbus Slave Address: 7
- Lines Per Rotation: 334
- Input Mode Byte : 00
- Input Control Byte : 00
- Speed: 9000 (with + and - buttons)
- Acceleration: 20000 (with + and - buttons)
- A purple text prompt: "Press Enter to Write Individual Parameter"
- Three buttons at the bottom: "Read Parameters", "Write Parameters", and "Reset Parameters".

SET A PARAMETER

- Pressing 'Enter' on text box will change the parameter in drive but it will not be saved permanently. 'Write Parameters' button must be clicked to save all parameters permanently.
- Set Position Proportional gain, Velocity Feed Forward gain and Position Integral Gain as per requirement. These can be used to remove vibrations and making movement of motor accurate.
- Set Modbus slave address and lines per rotation. Lines Per Rotation is very important because it does have effect on indication of speed feedback.
- If slave address is changed the software will automatically change the slave id of currently connected device.
- Set Speed and Acceleration as per the requirement of application.

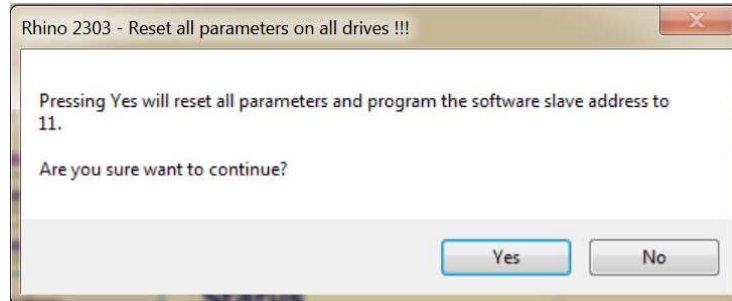
WRITE PARAMETERS

Once all the parameters are entered and tested, click on "Write Parameters" to save all parameters in drive. These parameters will be changed permanently. Notification will be shown as below when all parameters are written successfully.

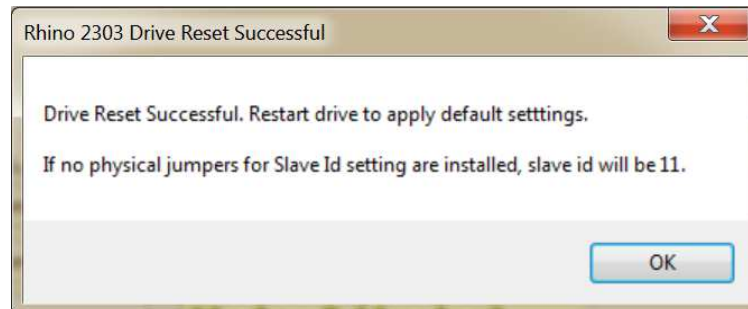


RESET PARAMETERS

To reset the drive to default configuration click 'Reset Parameters' button. This will set all parameters to default values. When button is clicked software will ask for confirmation.



Click on "Yes" to reset the drive. After Clicking on 'Yes' if reset is successful confirmation message will be shown. Power off and Power on the drive to load default parameters. If any physical jumpers are preset at this point the slave id of the drive will be according to that otherwise slave id will be 11.



Control Modes

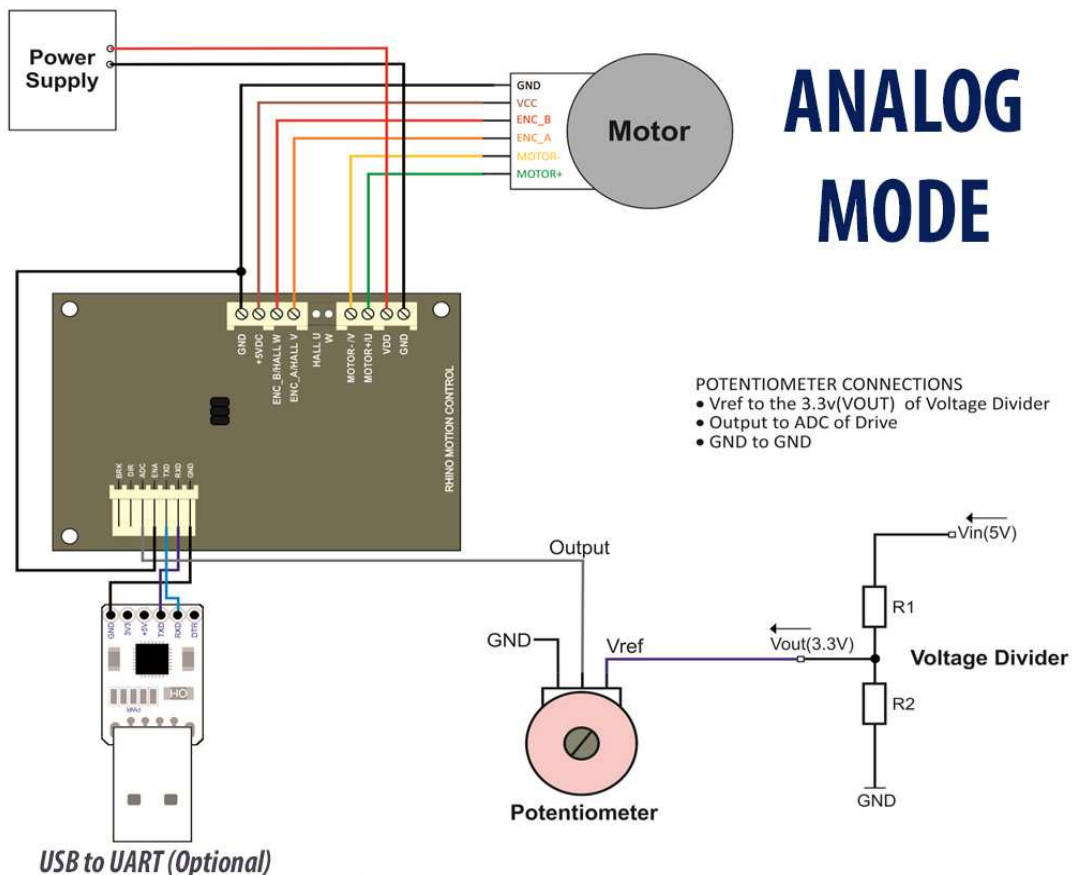
Motor can be run in 3 different modes - 0 : Analog Mode, 1 : Digital Speed Mode, 2 : Position Mode

Mode 0 - Analog Control Mode

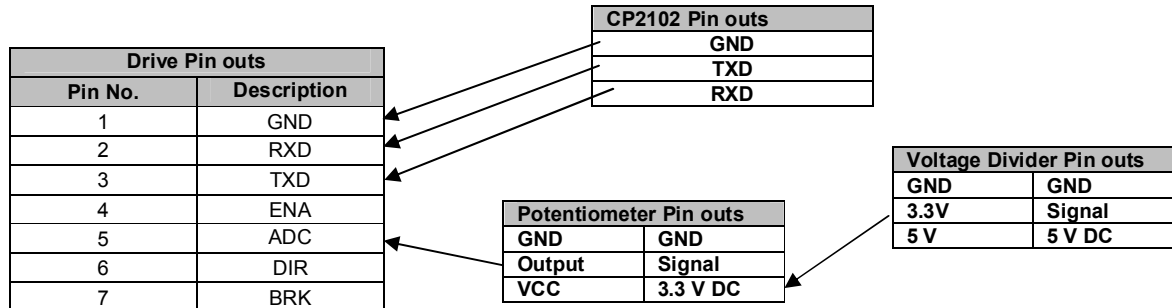
In this mode drive can be controlled using an external potentiometer or analog voltage input. As this is a close loop drive, high torque (up to the maximum torque motor can provide) is available even at low speed.

- Connect external potentiometer to change speed of motor as per diagram.
- Connect enable (Pin 4) of drive to GND (Pin 8) to enable motor in analog mode.
- Direction of motor can be changed in analog mode by connecting the Direction (Pin 6) to the GND (Pin 8).
- Electronic brake can be applied to motor by connecting Brake to GND - (Pin 7) to GND (Pin 1).

Hardware Connection:

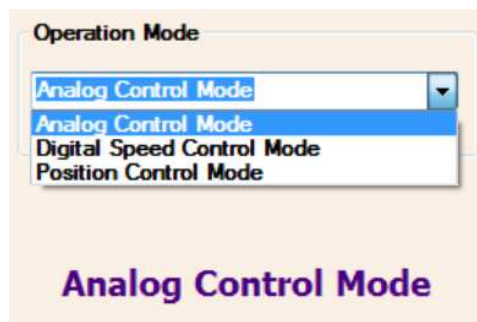


Drive to PC/Analog Connection



Set drive to Analog Control Of Motor in Software:

- By default the drive is in Analog mode. However if the mode is changed to some other then it can be set through software.
- Connect drive to PC and select **Analog Control Mode** in Operation Mode

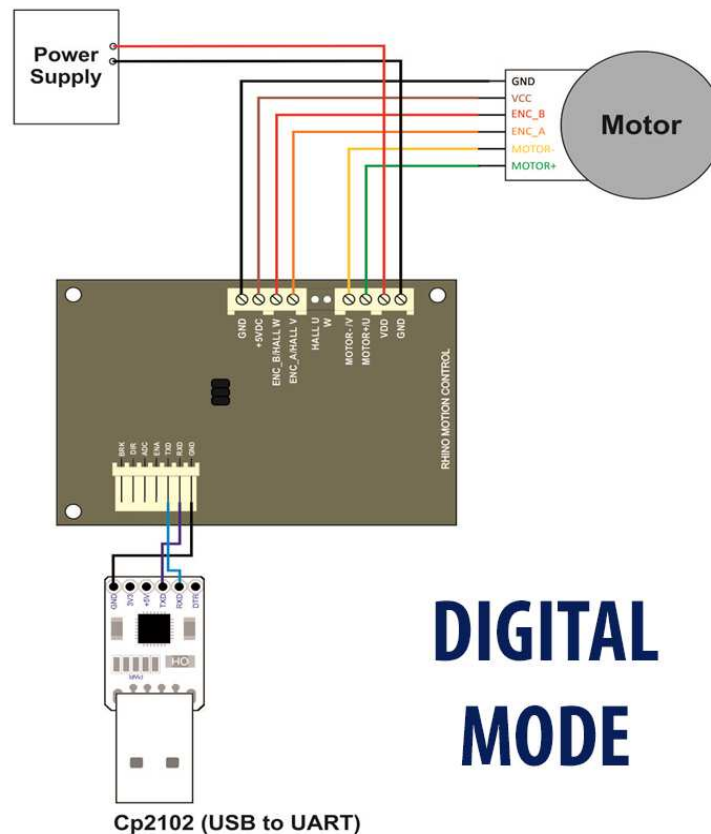


- After selecting analog control mode click on "Write Parameters" to always start the drive in analog mode.

DIGITAL SPEED CONTROL MODE

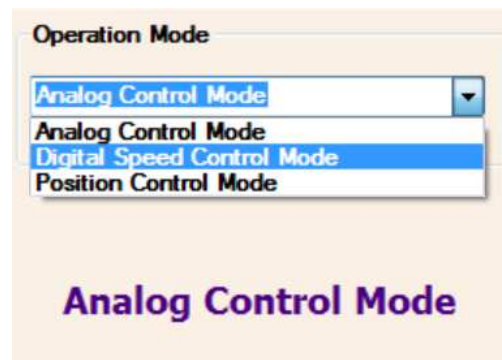
- In this mode the drive works in speed mode. It tries to reach and maintain the speed as per set parameter.
- The speed can vary depending upon load and gains set.
- Make sure that the set speed is always less than the motor can actually reach, otherwise the drive can give uncertain result. For example if the maximum speed motor can reach at given voltage is 18000 always set speed less than 18000 RPM.

Hardware Connection:



Digital Control Mode in Software:

- a). To run your motor on digital speed control mode select this mode from operation mode.



b) In this mode speed of motor(rpm) can be controlled by changing value from 0-65535.

The screenshot shows the Rhino Motion Controls software interface. It is divided into several sections:

- Parameters:** Includes fields for Position P Gain (32), FeedForward Gain (32), Position I Gain (16), Modbus Slave Address (7), Lines Per Rotation (334), Input Mode Byte (01), and Input Control Byte (00). There are also Speed (9000) and Acceleration (20000) fields with increment/decrement buttons. Buttons for 'Read Parameters', 'Write Parameters', and 'Reset Parameters' are present.
- Modbus Serial Port Connection:** Shows COM8 selected, 'Connected' status, Modbus Slave Address (7), Available Slaves (7), and Update Interval (50 ms).
- Operation Mode:** Set to 'Digital Speed Control Mode'.
- Speed Control:** Includes a 'Change Direction' button and a note to 'Change Speed and Acceleration from Parameters section'.
- Feedback:** Displays 'Current' (671027) and 'Position' (0).
- Mode:** Displays 'Mode : 1 (Speed)', 'Control : 00', and 'Motor Disabled'.
- Buttons:** 'Enable Motor' and 'Restart Drive' buttons are at the bottom.

Numbered annotations point to specific features:

1. Enter required speed value (points to the Speed field).
2. Enable motor (points to the 'Enable Motor' button).
3. change direction of motor (points to the 'Change Direction' button).
4. Current Speed of motor (points to the 'Speed' feedback display).

1. Speed can be increased and decreased using "+" and "-" respectively.

2. Enable Motor : Once speed is set click on "Enable Motor" to run motor on set speed.

3. Change Direction: Change the direction of motor. Can be done when motor is running or stopped.

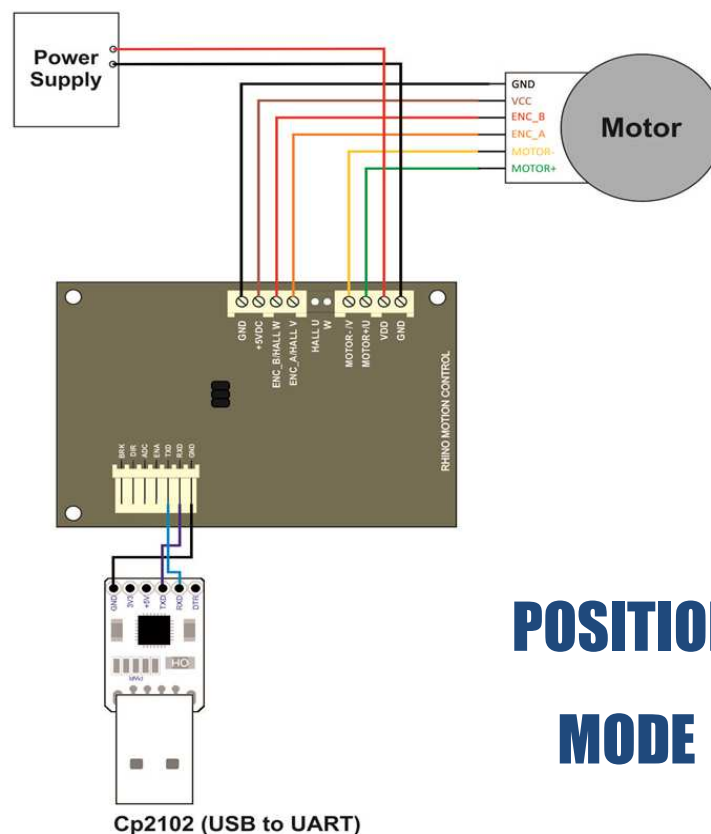
4. Speed Feedback: Current speed feedback from drive. This depends on actual speed measured by encoder and Lines per rotation setting.

POSITION CONTROL MODE

The position control mode allows to go to absolute position with reference to 0 as start position. Current position can be set to 0 at any point of time. The drive also manages the speed and acceleration/deceleration as per set parameter while responding to any change in set position.

When a 32bit value is set through register 16 and 18 the drive will try to move motor to that position. It can be updated even when motor is moving. Motor can be stopped any time or speed can be changed anytime even while moving.

Hardware Connection:



POSITION MODE

COUNT CALCULATION:

Lines per rotation concept is very important for Position Control mode. For example we will use HIGH TORQUE HIGH PRECISION ENCODER DC GEARED MOTOR 12V 200RPM (RMCS-5014)

<https://robokits.co.in/motors/encoder-dc-servo/high-torque-high-precision-encoder-dc-geared-motor-12v-200rpm>

Lines per rotation = 334

For one rotation of encoder, the no of counts = $334 \times 4 = 1336$ (As it is quad encoder)

So for 1336 steps, the base motor would move one rotation.

However as the motor is geared motor, the ratio of the gear box determines the rotation of the output shaft of the motor for each rotation of the base motor.

output shaft(rpm) = base motor(rpm) / gear ratio

counts per rotation = counts per rotation X gear ratio
for output shaft of base motor

For example, A motor of 200 RPM with base motor of 18000 rpm, the gear ratio is 90. Hence for a full rotation of the output shaft in this motor, the number of counts to be programmed would be $1336 \times 90 = 120,240$ steps.

This means output shaft will complete one rotation when Hex value of 120240 counts is set to register 16 and 18, LSB and MSB respectively.

Position Control in Software

When drive is connected and put in Position Control mode, software will look like this.

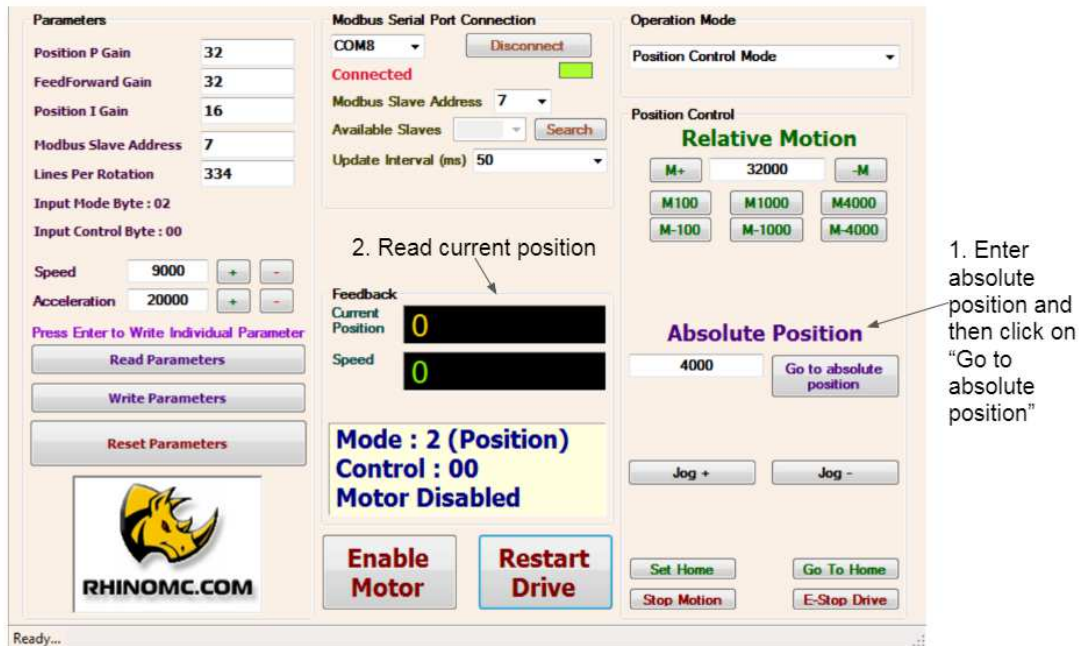
The screenshot shows the Rhino Motion Controls software interface. The 'Parameters' section on the left includes fields for Position P Gain (32), FeedForward Gain (32), Position I Gain (16), Modbus Slave Address (7), Lines Per Rotation (334), Input Mode Byte (02), and Input Control Byte (01). Below these are buttons for 'Read Parameters', 'Write Parameters', and 'Reset Parameters'. The 'Modbus Serial Port Connection' section shows 'COM8' selected and 'Connected' status. The 'Operation Mode' section shows 'Position Control Mode' selected. The 'Position Control' section has two sub-sections: 'Relative Motion' with buttons for M+, M100, M-100, 32000, M1000, M-1000, M4000, and M-4000; and 'Absolute Position' with a text box containing '4000' and a 'Go to absolute position' button. At the bottom, there are 'Jog +', 'Jog -', 'Set Home', 'Go To Home', 'Stop Motion', and 'E-Stop Drive' buttons. A status bar at the bottom left says 'Ready...'. The Rhino logo and 'RHINOMC.COM' are in the bottom left corner.

In Position control mode there are two ways to control position: **Absolute Position control** and **Relative Position control**.

ABSOLUTE POSITION

In this control mode motor will move precisely until entered count is not obtained. Position command can be given by putting number in Absolute Position textbox and clicking 'Go to absolute position' button. The direction is automatically managed by the drive. For example writing 4000 and clicking button moves the motor forward to 4000

count, giving -4000 would move the motor in other direction to achieve -4000 count. Feedback section shows the actual current position and speed of motor.



RELATIVE POSITION

Relative position is a software function, the software takes current position as reference and moves motor to the count typed in textbox with reference to current position.

For example, if current position is 5000 and entered value is 3000 in forward direction (M+) then motor will move to 8000 position (absolute) and then if direction is reverse (M-) then it will move to 2000 position (absolute).

There are also some preset command buttons to move motor. Any of these buttons will not function if motor is moving (speed is not 0).

Other commands in Position mode

"Jog+" and "Jog-" are used to move motor manually in forward and reverse direction. It can be used when motor needs to be manually moved to some location like for homing.

"Set Home": To set current position as home position - 0.

"Go to Home": Go to position 0.

"Stop Motion": Stop motor but keep position loop on. Motor stops but holds the current position.

"E-stop Drive": By clicking this button drive will cut-off power supply to motor and motor gets free.

Modbus Registers

Register	Function	Range / Command in HEX (Decimal)	Default value in HEX (Decimal)	Details
00	E-stop	0700 (1792)		Stops motion and stops giving power to motor
	Stop	0701 (1793)		Stops motion, motor maintains the position
	Set Home Position	0800 (2048)		Set current encoder count to 0
	Restart Drive	0900 (2304)		Restarts the drive
	Write Parameters to EEPROM	FF (255)		Write register is 2 bytes so write with slave id. If slave id is 7 - write 07FF
01	Modbus Slave Address	00 - F7 (0 - 247)	0B (11)	
02,03	Control and Mode byte		0000 (0)	
	Analog Control Mode	0001 (1)		Enable motor in analog Control mode
		0000 (0)		Disable motor in analog Control mode
	Digital Speed Control Mode	0101 (257)		Enable motor in CW
		0100 (256)		Disable motor in CW
		0104 (260)		Brake in CW
		0109 (265)		Enable motor in CCW
		0108 (264)		Disable motor in CCW
		010C (268)		Brake in CCW
	Position Control Mode	0201 (513)		Enable motor in position control mode
		0200 (512)		Disable motor in position control mode
04	Position Proportional Gain	01 - FF (1 - 255)	20 (32)	Writing 0 to this parameter and then saving parameters to eeprom loads system default values.
06	Position Integral Gain	00 - FF (0 - 255)	10 (16)	
08	Velocity Feed forward Gain	00 - FF (0 - 255)	20 (32)	
10	Lines per Rotation	0000 - FFFF (0 - 65535)	014E (334)	Set number of lines of encoder. Speed feedback depends on this.
12	Acceleration	0000 - FFFF (0 - 65535)	4E20 (20000)	

14	Speed Command for base Motor in RPM	0000 - FFFF (0 - 65535)	0800 (2048)	If gearbox is attached output RPM will be different.
16	LSB of Position Command	0000 - FFFF (0 - 65535)	0000 (0)	Send Position Command, motion is affected only when register 18 is updated.
18	MSB of Position Command	0000 - FFFF (0 - 65535)	0000 (0)	
20	LSB of Position Feedback	0000 - FFFF (0 - 65535)	0000 (0)	Position Feedback Register (Read Only)
22	MSB of Position Feedback	0000 - FFFF (0 - 65535)	0000 (0)	
24	Speed Feedback	0000 - FFFF (0 - 65535)	0000 (0)	Current Speed (Read Only)

Brief Description of all Registers:

Register 0: Special Commands

- **E-stop - 0X0700 (1792)**
 - This works like Emergency Stop. It stops motion immediately and also stops giving power to motor so motor becomes free.
- **Stop 0701 (1793)**
 - Stops motion with deceleration and maintains the position. The motor will work in close loop.
- **Set Home Position 0800 (2048)**
 - Set current encoder count to 0. This helps in achieving the relative motion. For example if a DC servo is used in linear slider application and on one side there is a limit switch to detect home position - this command can be used to set 0 position at the location.
- **Restart Drive 0900 (2304)**
 - This command restarts the drive. This is a software restart and simulates power cycle. Can be used to revert to EEPROM saved parameters.
- **Write Parameters to EEPROM FF (255)**
 - When any parameter is modified its not saved to EEPROM of drive so when drive restarts these parameters are lost. This commands is used to permanently save parameters to drive.
 - Writing register takes 2 bytes input so if only FF is written it also overwrites the address 1 which is slave id. So it's advisable to write with current slave id. If slave id is 1 then write 0X01FF.

Register 1: Modbus Slave Address (Read - Write)

- RMCS2303 Drive works on Modbus protocol. In Modbus protocol there is one master and several slaves communicate through standard way.
- For precise Communication each slave has unique address. Register 1 is allocated for slave address. maximum value for slave address is 247.
- This parameter is overridden if physical jumpers are present.

Register 2,3: Control and Mode byte (Read - Write)

- Register 3 is allocated for mode. as described earlier there are three different mode: Analog control(00), Digital Speed Control(01) and Position Control(02).
- Register 2 is control byte which is allocated for kind of motion. For example, to stop motor in Digital control mode while it is running in reverse direction 0x09 is to be set to register 2. It will stop motor with deceleration.
- All the modes with all controls are described in register table. These 2 registers are read and written together.
- Default value of register 2 and 3 is 0x0000.

Register 4: Position Proportional Gain (Read - Write)

- This register is used to set position proportional gain.
- Default value is 0X20 (32). Range is 1 to FF in hex and 1 to 255 in decimal.
- Writing 0 to this address and then sending write eeprom command (FF) on address 0 resets all parameters to default.

Register 6: Position Integral Gain (Read - Write)

- This register is used to set position Integral gain.
- Default value is 0X10 (16).
- Range is 0 to FF in hex and 0 to 255 in decimal.

Register 8: Velocity Feed forward Gain (Read - Write)

- This register is used to set Velocity feed forward gain.
- Default value is 0X20 (32).
- Range is 0 to FF in hex and 0 to 255 in decimal.

Register 10: Lines per Rotation (Read - Write)

- This register is allocated for Lines per rotation of an Encoder which is connected to motor.
- This is different than PPR or CPR. In a quadrature encoder Lines Per Rotation = PPR/4
- Default value is 0x014E (334).
- The speed feedback and speed command depends on this parameter.

Register 12: Set Acceleration (Read - Write)

- This register is for setting of acceleration.
- Default Hex value is 4E20 (20000).
- Acceleration is applied in speed and position control modes when starting the motion or changing the speed.
- Deceleration is also applied as per set value when stopping the motor or decreasing the speed.

Register 14: Set Speed (Read - Write)

- This register is used to set the speed of motor in RPM. Range is 0 to FFFF in hex and 0 to 65535 in decimal.

Register 16,18: Set Position (Read - Write)

- When drive is in position control mode, it will always try to achieve the position set in these registers.
- Position is a 32 bit unsigned value in which register 16 is LSB and 18 is MSB. The actual range of value is 0 to FFFFFFFF in hex and 0 to 4294967295 in decimal.
- On start up encoder count is always 0. Position is updated in all modes.
- Drive applies changes only when register 18 is updated.
- Direction will be chosen automatically by the drive to achieve the position.

Example 1. Desired position is 250000 then convert 250000 in hex and which is 0x3D090. These registers are 16 bit so send LSB of Hex value(0xD090) in register 16 and MSB of Hex value(0x0003) in register 18.

Example 2. Desired position is -1000 then calculate counts as described below:

Position = 4294967296 -1000 = 4294966296

Then convert count value in Hex which is 0xFFFFFC18. Now send LSB (FC18) in register 16 and MSB(FFFF) in register 18.

Converted values can also be seen in software in status bar when giving any motion command.



Register 20,22: Position Feedback

- These registers are used to read current position of encoder.
- Register 20 is LSB and register 22 is MSB.
- It's a 32 bit unsigned number. Range is 0 to FFFFFFFF in hex and 0 to 4294967295 in decimal.
- This number can be converted to signed number to know forward or reverse motion - pc software uses same method.
- Positive range is 1 to 2147483647 and negative range is -1 to -2147483648.

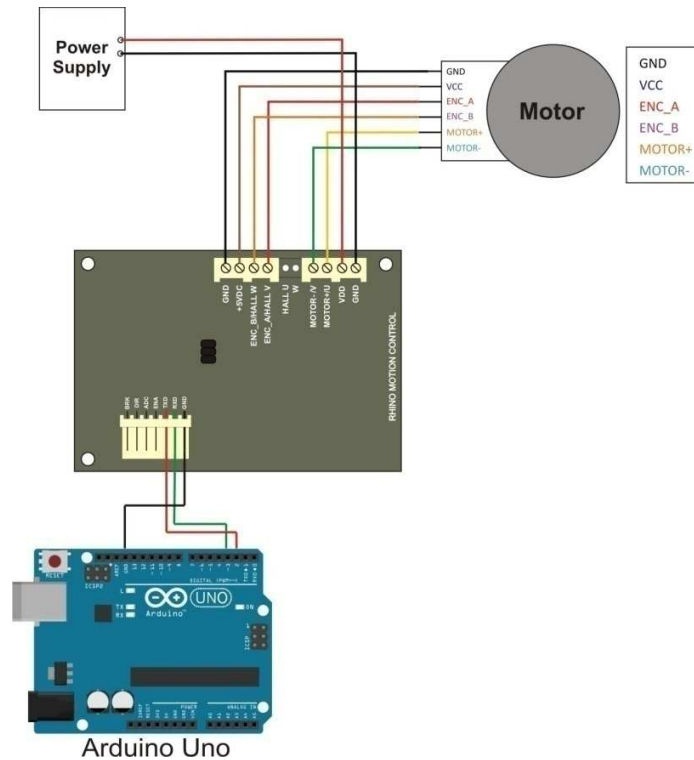
Register 24: Speed Feedback

Register 24 is used for Speed feedback. Current speed of motor in RPM can be read from this register. This value depends on Lines per rotation set value. This value is signed and negative value means speed in reverse direction.

Control Through Arduino Microcontroller

RMCS2303 drive can be controlled by any microcontroller, PLC or PC software capable of communicating on ASCII Modbus protocol. Following section shows how to use provided library for Arduino microcontroller.

Hardware Connection:



DRIVE-ARDUINO MEGA CONNECTION

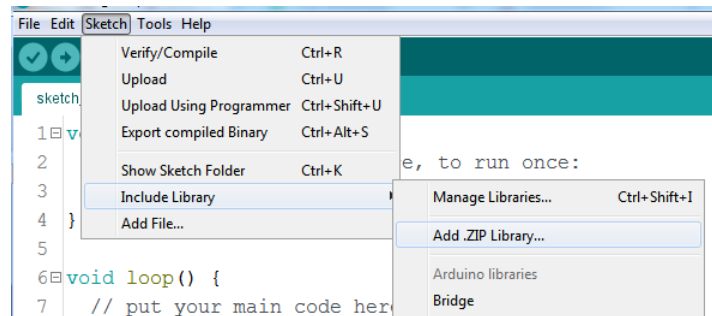
DRIVE	ARDUINO
GND	GND
RXD	TXn (n=1,2,3 for Arduino mega2560)
TXD	RXn (n=1,2,3 for Arduino mega2560)

DRIVE-ARDUINO UNO CONNECTION

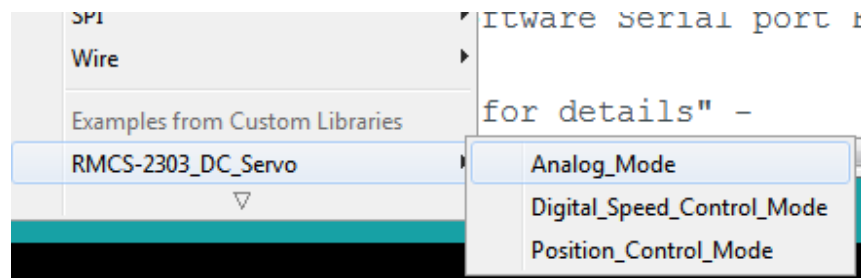
DRIVE	ARDUINO
GND	GND
RXD	D3(Software Serial Tx)
TXD	D2(Software Serial Rx)

Library Installation:

- Download RMCS-2303 Arduino library from :
http://robokits.co.in/downloads/RMCS2303drive_Arduino_Library.zip
- In Arduino IDE got to menu - Sketch > Include Library > Add .ZIP Library



- Choose the downloaded zip file.
- You can see confirmation if library import is successful
- Library added to your libraries. Check "Include library" menu
- Open examples from Menu File > Examples > RMCS-2303_DC_Servo



- Set parameters, make sure you choose parameters, serial port, slave id properly before uploading the code to Arduino board.
- There are many more functions other than covered in example codes. Description of these functions as below. In this code 'rmcs' is object for library class, it can be changed.

Functions for analog control mode

```
rmcs.Enable_Analog_Mode(slave_id); //Enable analog mode
rmcs.Disable_Analog_Mode(slave_id); //Disable motor in analog mode
```

Functions for Digital speed control mode

```
rmcs.Speed(slave_id,5000); //Set speed to 5000 RPM
rmcs.Enable_Digital_Mode(slave_id,1); //Enable speed mode, motor starts moving
long int b=rmcs.Speed_Feedback(slave_id); //Speed feedback from drive
rmcs.Brake_Motor(slave_id, 1); //Brake motor in speed mode
rmcs.Disable_Digital_Mode(slave_id,1); //Disable motor in speed mode
```

Functions for position control mode

```
mcs.Absolute_position(slave_id,-10000);    //Move to absolute position -10000
long int b=mcs.Position_Feedback(slave_id);//Take position feedback from drive
Serial.println(b);
```

Special Functions

```
mcs.SAVE(slave_id);    //Save all parameters
mcs.RESET(slave_id);   //Reset drive to default parameters,
mcs.ESTOP(slave_id);   //E-stop the drive
mcs.STOP(slave_id);    //Stop motion
mcs.SET_HOME(slave_id);    //Set current position to 0
mcs.Restart(slave_id);    //Software restart drive.
```

Description of code and functions used

```
#include<RMCS2303drive.h>
```

Include Downloaded library in code to use available functions for controlling RMCS2303 drive in all modes. if library is not installed then follow steps as per described in " Library Installation".

```
RMCS2303 mcs;
```

Make an object for class RMCS2303. Any object name can be set instead of "mcs". Use the same class in all functions.

```
byte slave_id=7; int INP_CONTROL_MODE=265; int PP_gain=32; int PI_gain=16; int VF_gain=32;  
int LPR=334; int acceleration=20000; int speed=320;
```

Set all necessary parameters value as per application. **INP_CONTROL_MODE** depends on Mode and type of control, values are given in Hex in Appendix 1 so convert it in integer then store in **INP_CONTROL_MODE** variable. **LPR** is Lines Per Rotation.

```
mcs.Serial_selection(1);    //Serial port selection:0-Hardware serial,1-Software serial
```

Set hardware or software serial port to be used. Boards like Arduino Uno have only one hardware serial port which gets used by usb-serial. In this case software serial should be used to communicate with drive. Boards like Arduino mega have multiple hardware serial ports, in this case use hardware serial.

Hardware Serial port: **mcs.Serial_selection(0);**

Software Serial Port: **mcs.Serial_selection(1);**

```
mcs.Serial0(9600);    //usb serial to monitor data on serial monitor
```

This function will initialize Serial communication by setting baudrate between Arduino and Serial monitor in Arduino IDE. Baudrate can be changed as required.

```
rmcs.begin(&Serial3,9600); //uncomment if using hardware serial port for  
                           mega2560:Serial1,Serial2,Serial3 and set baudrate
```

This function passes the pointer of serial port to be used. It's used to initialize serial port and set baudrate.

```
Serial1 : rmcs.begin(&Serial1,9600);
```

```
Serial2 : rmcs.begin(&Serial2,9600);
```

```
Serial3 : rmcs.begin(&Serial3,9600);
```

```
rmcs.begin(&myserial,9600); //uncomment for using software serial and set baudrate
```

This function is used to set baudrate of Software Serial port. In Arduino Uno software serial should be used.

```
rmcs.WRITE_PARAMETER(slave_id, INP_CONTROL_MODE, PP_gain, PI_gain, VF_gain, LPR,  
acceleration, speed);
```

This function is used to write all parameters. Confirmation will be sent on Serial port shown as below when all parameters are written Successfully.

```
INP_CONTROL :      DONE  
INP_MODE :         DONE  
PP_GAIN :          DONE  
PI_GAIN :          DONE  
VF_GAIN :          DONE  
LPR :              DONE  
ACCELERATION :     DONE  
SPEED :            DONE
```

```
rmcs.READ_PARAMETER(slave_id);
```

This function read all current parameters and then print on Serial monitor.

Analog Mode:

```
rmcs.Enable_Analog_Mode(slave_id);
```

This function will enable your drive in Analog control mode. Speed of motor can be changed by varying potentiometer.

```
rmcs.Disable_Analog_Mode(slave_id); // To disable motor in Analog control Mode
```

This function is used to disable motor in analog control mode.

Digital Speed Control Mode:

```
rmcs.Speed(slave_id,5000); //enter speed within range of 0-65535
```

When drive is used in Digital Speed control mode, this function is used to change the speed. Value from 0 to 65535 can be set but all motors have some maximum rpm. When value is more than possible RPM motor will run at full speed, this may cause jerks in motion commands.

```
rmcs.Enable_Digital_Mode(slave_id,1);
```

Enable motor in digital mode with forward or reverse direction. 0 is forward (encoder count +) and 1 is reverse (encoder count -).

Forward : `rmcs.Enable_Digital_Mode(slave_id,0);`

Reverse: `rmcs.Enable_Digital_Mode(slave_id,1);`

```
rmcs.Speed_Feedback(slave_id);
```

This function returns the current speed of motor in long integer. Speed less than 0 is received when motor is moving in reverse direction.

```
rmcs.Brake_Motor(slave_id, 1);
```

This function is used to brake motor in digital speed control mode. Should be used with direction as below, if not drive may misbehave and motor may jerk.

Forward: `rmcs.Brake_Motor(slave_id,0);`

Reverse: `rmcs.Brake_Motor(slave_id,1);`

```
rmcs.Disable_Digital_Mode(slave_id,1);
```

This function will disable Digital speed control mode in entered direction. Should be used with direction as below, if not drive may misbehave and motor may jerk.

Forward: `rmcs.Disable_Digital_Mode(slave_id,0);`

Reverse: `rmcs.Disable_Digital_Mode(slave_id,1);`

Position Control Mode:

```
rmcs.Absolute_position(slave_id,10000);
```

This function enables drive in position control mode. Positive range is 1 to 2147483647 and negative range is -1 to -2147483648. Direction is selected automatically by drive.

`rmcs.Absolute_position(slave_id,-10000);`

```
rmcs.Position_Feedback(slave_id);
```

Current Position of motor can be read by this function on Serial monitor. This function will return long integer.

Special Functions:

rmcs.SAVE(slave_id);

This function will save all written parameters in drive.

rmcs.RESET(slave_id);

This function will reset all the parameters and set all parameters at default value.

rmcs.ESTOP(slave_id);

This function will stop motor and cut off power from drive to motor.

rmcs.STOP(slave_id);

This function is used to stop motion of motor. Motor will be locked as its still controlled by drive.

rmcs.SET_HOME(slave_id);

This function changes current encoder position to "0".

rmcs.Restart(slave_id);

Restart the drive by software.